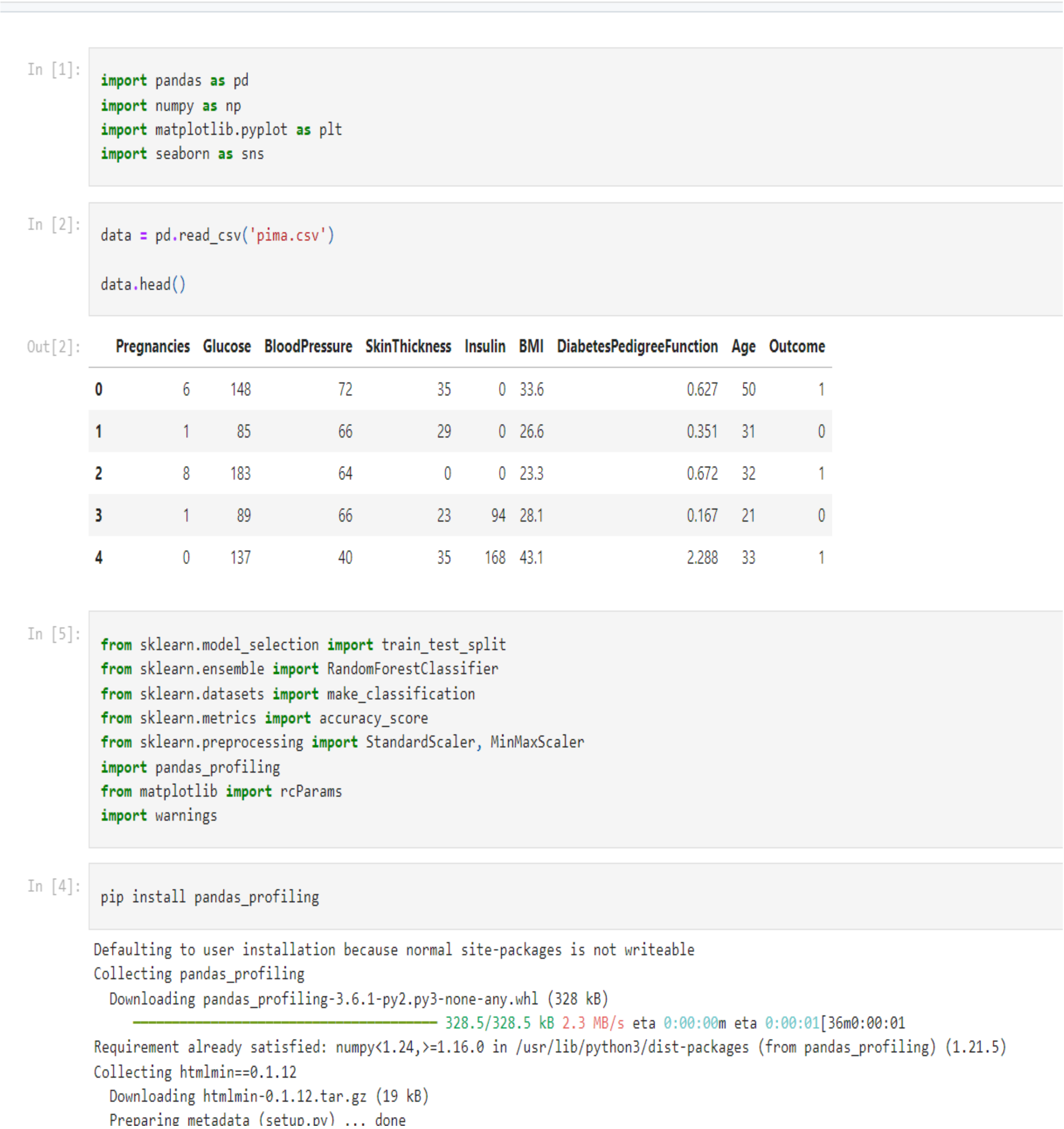
RANDOM FOREST REGRESSION



In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
data = pd.read_csv('pima.csv')

data.head()
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [5]:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import pandas_profiling
from matplotlib import rcParams
import warnings
```

In [4]:

```
pip install pandas_profiling
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting pandas_profiling
  Downloading pandas_profiling-3.6.1-py2.py3-none-any.whl (328 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 328.5/328.5 kB 2.3 MB/s eta 0:00:00m eta 0:00:01[36m0:00:01
Requirement already satisfied: numpy<1.24,>=1.16.0 in /usr/lib/python3/dist-packages (from pandas_profiling) (1.21.5)
Collecting htmlmin==0.1.12
  Downloading htmlmin-0.1.12.tar.gz (19 kB)
  Preparing metadata (setup.py) ... done
```

```
Note: you may need to restart the kernel to use updated packages.
```

In [6]:

```
data.columns
```

Out[6]:

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [7]:

```
X=data.drop("Outcome",axis=1)
y=data["Outcome"]
```

In [8]:

```
scaler=StandardScaler()
X_scaled=scaler.fit_transform(X)

X_train,X_test,Y_train,Y_test=train_test_split(X_scaled,y,stratify=y,test_size=0.10,random_state=34)
```

In [9]:

```
classifier = RandomForestClassifier(n_estimators=100)
classifier.fit(X_train,Y_train)
```

Out[9]:

```
RandomForestClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

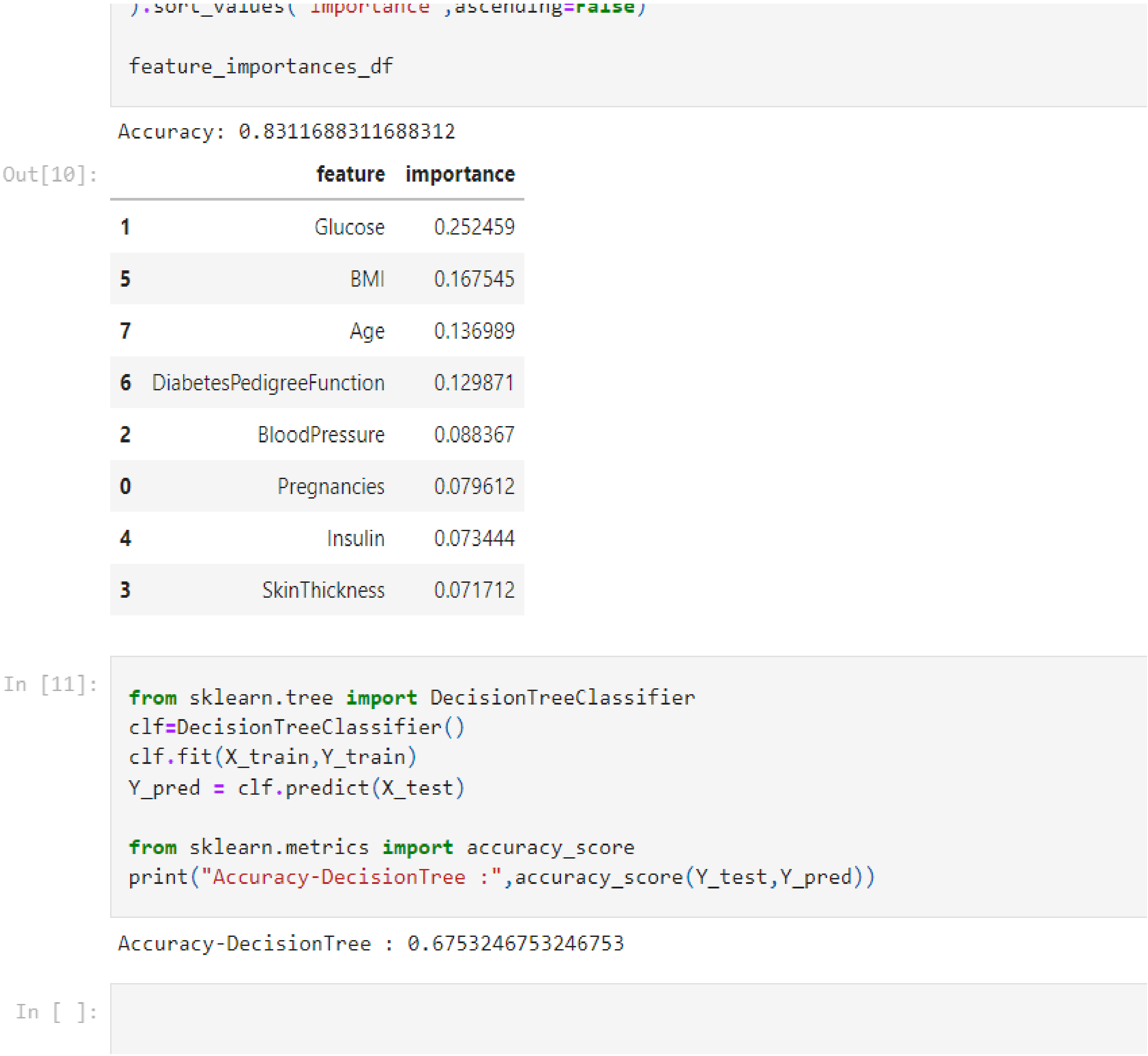
In [10]:

```
y_pred = classifier.predict(X_test)

print("Accuracy:",accuracy_score(Y_test,y_pred))

feature_importances_df = pd.DataFrame(
    {"feature":list(X.columns),"importance":classifier.feature_importances_}
).sort_values("importance",ascending=False)

feature_importances_df
```

```
Accuracy: 0.8311688311688312
```

Out[10]:

	feature	importance
1	Glucose	0.252459
5	BMI	0.167545



```
).sort_values('importance',ascending=False)

feature_importances_df
```

```
Accuracy: 0.8311688311688312
```

Out[10]:

	feature	importance
1	Glucose	0.252459
5	BMI	0.167545
7	Age	0.136989
6	DiabetesPedigreeFunction	0.129871
2	BloodPressure	0.088367
0	Pregnancies	0.079612
4	Insulin	0.073444
3	SkinThickness	0.071712

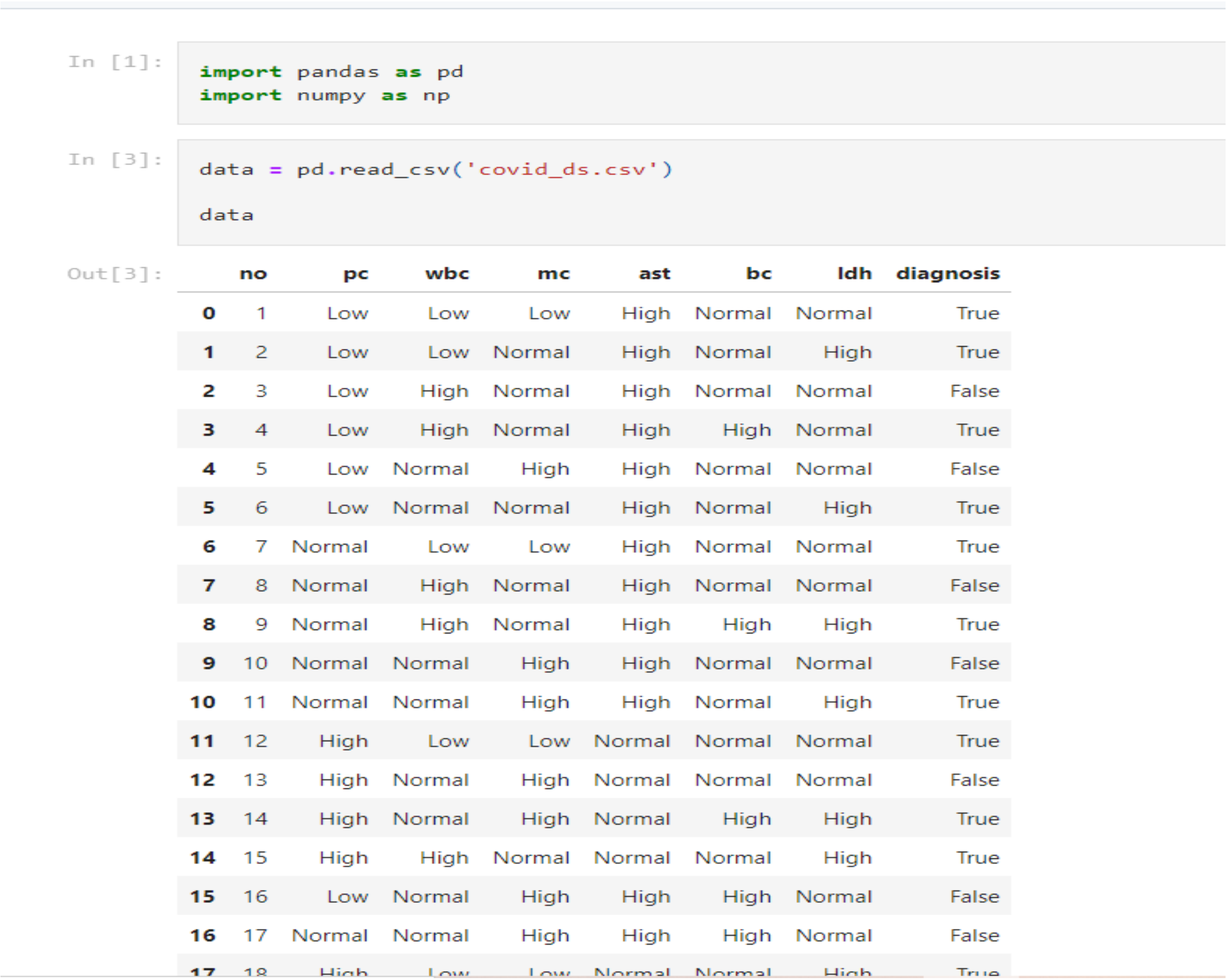
In [11]:

```
from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier()
clf.fit(X_train,Y_train)
Y_pred = clf.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy-DecisionTree :",accuracy_score(Y_test,Y_pred))
```

```
Accuracy-DecisionTree : 0.6753246753246753
```

In [ ]:

NAVIE BAYES



```
In [1]:  import pandas as pd
         import numpy as np
```

```
In [3]:  data = pd.read_csv('covid_ds.csv')

         data
```

```
Out[3]:     no      pc     wbc      mc     ast      bc     ldh  diagnosis
         0   1     Low     Low     Low    High  Normal  Normal       True
         1   2     Low     Low  Normal    High  Normal    High       True
         2   3     Low    High  Normal    High  Normal  Normal      False
         3   4     Low    High  Normal    High    High  Normal       True
         4   5     Low  Normal    High    High  Normal  Normal      False
         5   6     Low  Normal  Normal    High  Normal    High       True
         6   7  Normal     Low     Low    High  Normal  Normal       True
         7   8  Normal    High  Normal    High  Normal  Normal      False
         8   9  Normal    High  Normal    High    High    High       True
         9  10  Normal  Normal    High    High  Normal  Normal      False
        10  11  Normal  Normal    High    High  Normal    High       True
        11  12    High     Low     Low  Normal  Normal  Normal       True
        12  13    High  Normal    High  Normal  Normal  Normal      False
        13  14    High  Normal    High  Normal    High    High       True
        14  15    High    High  Normal  Normal  Normal    High       True
        15  16     Low  Normal    High    High    High  Normal      False
        16  17  Normal  Normal    High    High    High  Normal      False
        17  18    High     Low     Low  Normal  Normal    High       True
```

14	15	High	High	Normal	Normal	Normal	High	True
15	16	Low	Normal	High	High	High	Normal	False
16	17	Normal	Normal	High	High	High	Normal	False
17	18	High	Low	Low	Normal	Normal	High	True
18	19	Normal	Normal	Normal	High	Normal	Normal	False
19	20	Normal	High	Normal	High	Normal	High	True
20	21	Normal	Low	Normal	High	Normal	High	True
21	22	Low	High	Normal	High	High	High	True
22	23	Low	Low	Low	High	High	High	True
23	24	High	High	Normal	Normal	Normal	Normal	True
24	25	High	Normal	Normal	Normal	Normal	Normal	False

In [4]:

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
pc_encoded=le.fit_transform(data['pc'].values)
wbc_encoded=le.fit_transform(data['wbc'].values)
mc_encoded=le.fit_transform(data['mc'].values)
ast_encoded=le.fit_transform(data['ast'].values)
bc_encoded=le.fit_transform(data['bc'].values)
ldh_encoded=le.fit_transform(data['ldh'].values)
Y=le.fit_transform(data['diagnosis'].values)
```

In [5]:

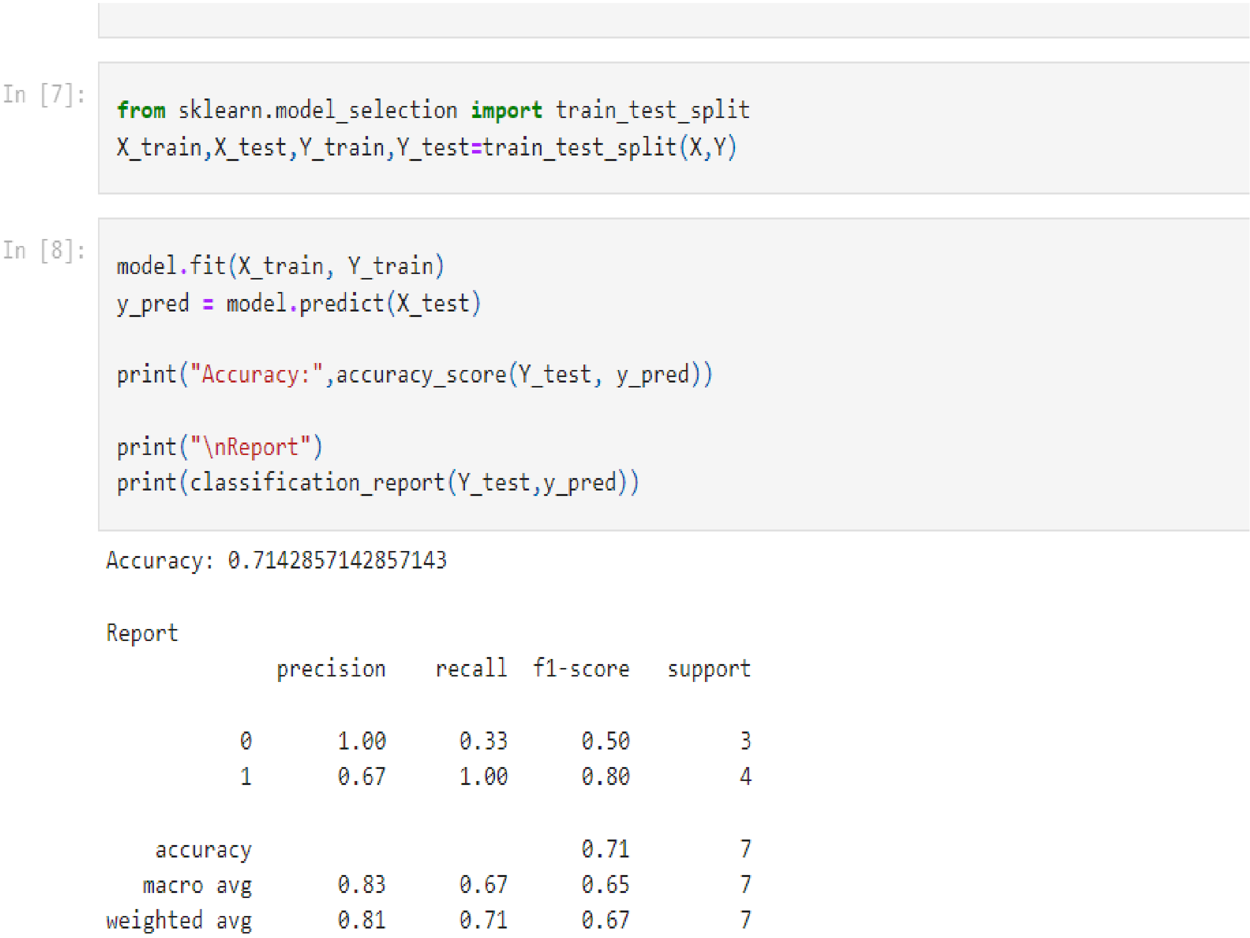
```
X=np.array(list(zip(pc_encoded,wbc_encoded,mc_encoded,ast_encoded,bc_encoded,ldh_encoded)))
X
Y
```

Out[5]:

```
array([1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1,
       1, 1, 0])
```

In [6]:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
model = MultinomialNB()
```



In [7]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y)
```

In [8]:

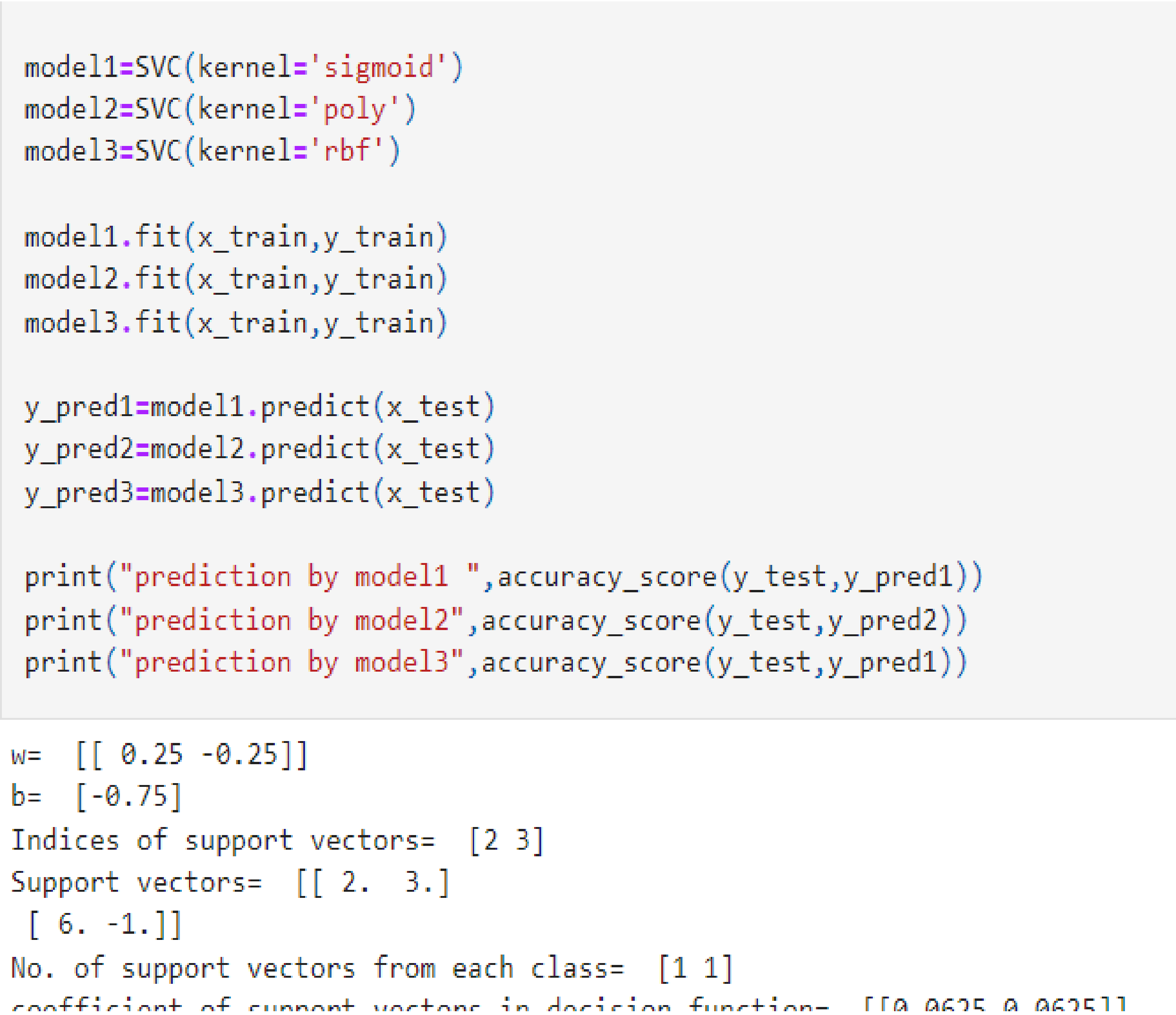
```
model.fit(X_train, Y_train)
y_pred = model.predict(X_test)

print("Accuracy:",accuracy_score(Y_test, y_pred))

print("\nReport")
print(classification_report(Y_test,y_pred))
```

```
Accuracy: 0.7142857142857143

Report
              precision    recall  f1-score   support

           0       1.00      0.33      0.50         3
           1       0.67      1.00      0.80         4

    accuracy                           0.71         7
   macro avg       0.83      0.67      0.65         7
weighted avg       0.81      0.71      0.67         7
```

SVM

In [2]:

```
from sklearn.svm import SVC
from sklearn import svm
import numpy as np

X=np.array([[3,4],[1,4],[2,3],[6,-1],[7,-1],[5,-3]])
y=np.array([-1,-1,-1,1,1,1])

l=SVC(C=1e5,kernel='linear')
l.fit(X,y)

print('w= ',l.coef_)
print('b= ',l.intercept_)
print('Indices of support vectors= ',l.support_)
print('Support vectors= ',l.support_vectors_)
print('No. of support vectors from each class= ',l.n_support_)
print('coefficient of support vectors in decision function= ',np.abs(l.dual_coef_))


import pandas as pd
data=pd.read_csv('glass.csv')
data.head()

x=data.drop('Type',axis=1)
y=data.Type

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)

linear=svm.SVC(kernel='linear')
linear.fit(x_train,y_train)

print(linear.support_vectors_)

print(linear.n_support_)

y_pred=linear.predict(x_test)

from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,y_pred))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test,y_pred))

from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```



```
model1=SVC(kernel='sigmoid')
model2=SVC(kernel='poly')
model3=SVC(kernel='rbf')

model1.fit(x_train,y_train)
model2.fit(x_train,y_train)
model3.fit(x_train,y_train)

y_pred1=model1.predict(x_test)
y_pred2=model2.predict(x_test)
y_pred3=model3.predict(x_test)

print("prediction by model1 ",accuracy_score(y_test,y_pred1))
print("prediction by model2",accuracy_score(y_test,y_pred2))
print("prediction by model3",accuracy_score(y_test,y_pred1))
```

```
w=  [[ 0.25 -0.25]]
b=  [-0.75]
Indices of support vectors=  [2 3]
Support vectors=  [[ 2.  3.]
 [ 6. -1.]]
No. of support vectors from each class=  [1 1]
coefficient of support vectors in decision function=  [[0.0625 0.0625]]
```